**Lab 2.** Simple Sentiment Analysis.

**Лабораторная работа 2.** Простейший анализ тональности текста.

**Цели:**

- окончательное закрепление навыков работы с базовыми коллекциями и конструкциями в языке Python

- приобретение навыков работы с различными сторонними модулями (библиотеками), такими как NumPy, Nltk, Matplotlib.

**Задачи:**

Главной задачей данной лабораторной работы является анализ и классификация тональности текста. Объектом наших исследований будет выборка небольших текстов-сообщений из Twitter по определенному событию/объекту.

Работа по лабораторной разбита на несколько этапов:

* Подготовка и обработка данных
* Частотный анализ
* Эмпирическая оценка/маркировка отдельных слов
* Сравнение правил оценки отдельных твитов
* Работа с частями речи
* Оценка распределения твитов по времени
* Дополнительные задания

Наш анализ не будет иметь серьёзных претензий на релевантность (для более релевантной оценки требуется достаточно объёмная выборка текстов и более серьёзные правила их оценки), однако, позволит нам достигнуть учебных целей и хорошо провести время.

**Данные:**

Данные представляют из себя твиты на русском языке, посвященные прошедшей жеребьевки ЧМ 2018. Данные содержатся в файле tweets.txt, твиты разделены между собой строкой «\*\*\*\*\*\*\*\*\*\*». В каждом твите помимо самого сообщения содержится информация о времени опубликования.   
NB: некоторые твиты могут оказаться пустыми (только информация о времени, без самого сообщения), воспринимайте их как выбросы и игнорируйте.

**1. Подготовка и обработка данных.**

Первый этап нашей работы является типичным для множества алгоритмов анализа данных – это этап первичной обработки данных. В качестве входных данных у нас имеется набор твитов. Назовём их “сырыми” (raw).

Каждый “сырой” твит может содержать различные элементы, которые неинтересны для нашего анализа и более того могут его затруднить и сделать менее релевантным, например:

- Вспомогательные части речи - предлоги, частицы.  
В большинстве случаев предлоги, частицы будут наиболее часто встречаемыми в тексте.  
Но несут ли они много информации для нас? Нет. Рассмотрение их наравне со значащими частями текстов может только затруднить нам работу и зря потратить вычислительные мощности.

- Специфические конструкции, присущие твитам.

Например, сочетание RT (ретвит) также часто используется в твитах, но, как и предлог, смысловой нагрузки, которая могла бы быть нам полезна, не несёт.

- Ссылки**.** Ещё один часто встречаемый элемент в текстах Twitter - ссылки (как правило, укороченные). Они часто прикрепляются к твитам. Для нашей задачи каждая такая ссылка может быть расценена как уникальное слово. Однако, оценить эмоциональную окраску такого слова не представляется возможным.

- Цифры, знаки препинания, специальные символы ($,%,-).

Данные элементы также не будут рассматриваться в нашем анализе.

Для каждой из перечисленных групп необходимо предложить вариант очистки текстов от данных элементов, для некоторых (например, предлоги и частицы) возможно их присутствие в тексте, однако, в этом случае они не должны оказывать влияния на оценку текста (например, помечены как “нейтральные”).

**2. Частотный анализ.**

После очистки данных от ненужных нам конструкций необходимо выполнить частотный анализ оставшейся информации.

В файл frequency.txt для каждого слова необходимо вывести количество твитов, в котором слово встречается и их процент от общего числа твитов.

Пример формата вывода:

good - 115 - 10%

bad - 10 - 0.1%

Trump - 1 - 0.01%

Каждая строка в файле frequency.txt – это слово, дальше через тире количество его вхождений во все твиты, далее через тире процент от всех твитов. Слова должны выводиться по убыванию частоты, то есть в первой строке – самое частое встречаемое слово, далее – второе по частоте и т.д.

Выполняя частотный анализ, вы можете заметить, что в вашем тексте по-прежнему присутствуют различные вспомогательные конструкции из этапа 1 (например, ссылки, начинающиеся не с http://, а с https://). В этом случае рекомендуем также обработать такие конструкции и заново выполнить частотный анализ.

**3. Эмпирическая оценка/маркировка отдельных слов.**

Далее необходимо создать файл с Вашей личной оценкой по каждому из слов в списке frequency.txt.

Каждое из слов необходимо оценить в рамках положительного/отрицательного/нейтрального смысла и поставить соответствующую метку:

1 - положительный смысл  
-1 - отрицательный смысл  
0 - нейтральный смысл/зависит от контекста

Оценки для каждого слова (из списка frequency.txt) необходимо записать в файл estimations.txt.

Пример:

good 1  
bad -1  
Trump 0

Каждая строка в файле estimations.txt это слово и через пробел его оценка.

Слов в списке может быть достаточно много (несколько сотен). Это задание займёт у Вас некоторое время (от 30 минут до 1 часа). Маркируйте слова мгновенно, для каждого из них (так как это в большинстве своем – слова русского языка) смысл и эмоциональная окраска очевидны.  
Не стоит задумываться о правильности/неправильности своего варианта, в данном случае важна именно Ваша оценка, так что правильного/неправильного варианта маркировки не существует.

**4. Выбор правила оценки твита. Оценка твитов.**

Далее Вам необходимо выбрать правило по которому Вы будете оценивать твит. Мы уже имеем информацию о каждом отдельном слове (из предыдущего шага), осталось выбрать как агрегировать эту информацию для оценки всего твита.

Можно использовать различные правила, например:

1. Сумма оценок. Просто суммируем оценки слов в твите и выносим свой вердикт на основе общей суммы.

2. Подсчёт количества или доли положительных/отрицательных/нейтральных слов. Например, если в твите одно “отрицательное” слово и 20 “нейтральных”, мы можем оценить его как “нейтральный” несмотря на отрицательную сумму. В остальном, правило практически идентично правилу суммы, так что придумайте что-нибудь своё 😊

Любые правила, которые Вы сможете придумать могут быть использованы (кроме тривиальных правил, например: всё – “положительно”, всё – “отрицательно” и т.д.).  
Наша задача в этом пункте – определить два правила, классифицировать твиты по каждому из них (то есть, для каждого правила всем твитам присвоить оценки -1, 0 или 1).

Далее мы сравним наши классификации и запишем результаты в файл classifications.txt.

Пример:

Summ rule  
Good - 1 - 0.01%  
Bad - 997 - 99.97%  
Neutral - 2 - 0.02%

Some rule 2  
Good - 700 - 70%  
Bad - 200 - 20%  
Neutral - 100 - 10%

Первая строка – имя правила (придумайте сами).  
Далее – три строки, которые описывают количество твитов отнесенных правилом к хорошим, плохим и нейтральным.  
Далее – пустая строка.  
Далее – второе правило по такому же шаблону.

Также необходимо построить график-гистограмму для каждого из выбранных правил.

**5. Части речи.**

С помощью библиотеки необходимо определить наиболее встречаемые 3 положительных и 3 отрицательных прилагательных, которыми описывается наша ситуация/объект в твитах, записать результаты в файл adjectives.txt.

Пример:

Top-5 Positive:  
захватывающий - 100 - 10%  
добрый - 100 - 10%  
хороший - 100 - 10%  
милый - 99 - 9.9%  
няшный - 2 - 0.02%

Top-5 Negative:  
плохой - 100 - 10%  
ужасный - 99 - 9.9%  
отвратительный - 99 - 9.9%  
болезненный - 2 - 0.02%  
нэвэльный - 2- 0.02%

Первая строка в файле adjectives.txt – заголовок “ Top-5 Positive:”.  
Далее идёт 5 строк, каждая из которых содержит слово, далее через тире количество твитов, в которых оно встречается, далее через тире процент от общего числа твитов.  
Далее идет пустая строка и аналогичная конструкция для “Top-5 Negative:”.  
Слова в обоих списках должны выводиться по убыванию частоты, то есть в первой строке – самое частое встречаемое слово, далее – второе по частоте и т.д.

Также необходимо построить сдвоенный график-гистограмму по данной информации.

**6. Оценить распределение положительных/отрицательных/нейтральных твитов по времени.**

Перед каждым твитом во входных данных находится информация о времени публикации.

Необходимо оценить распределение положительных/отрицательных/нейтральных твитов по часам, записать результаты в файл hours.txt и построить график динамики распределения по классам во времени.

Выбираем размер временного окна (ex. 30 мин) и шага (ex. 10 мин). На первом шаге выбираем все твиты попавшие во временное окно (т.е. в первые 30 мин) и рассчитываем для них распределение по классам (т.е. долю твитов кадого класса среди этих твитов). На втором и последующих шагах увеличиваем окно на шаг (т.е. берем уже 40 мин, потом 50 мин и т.д.), и снова рассчитываем распределение. Результаты необходимо записать в файл hours.txt в следующем формате:

Start - End : N N+/N0/N-

где Start и End – начало и конец временного окна; N – число твитов, попавших в окно; N+, N0, N- - доля положительных, нейтральных и отрицательных твитов соответственно.

Пример:

16:30 – 17:00 : 100 0.5/0.3/0.2

16:30 – 17:10 : 110 0.55/0.25/0.2

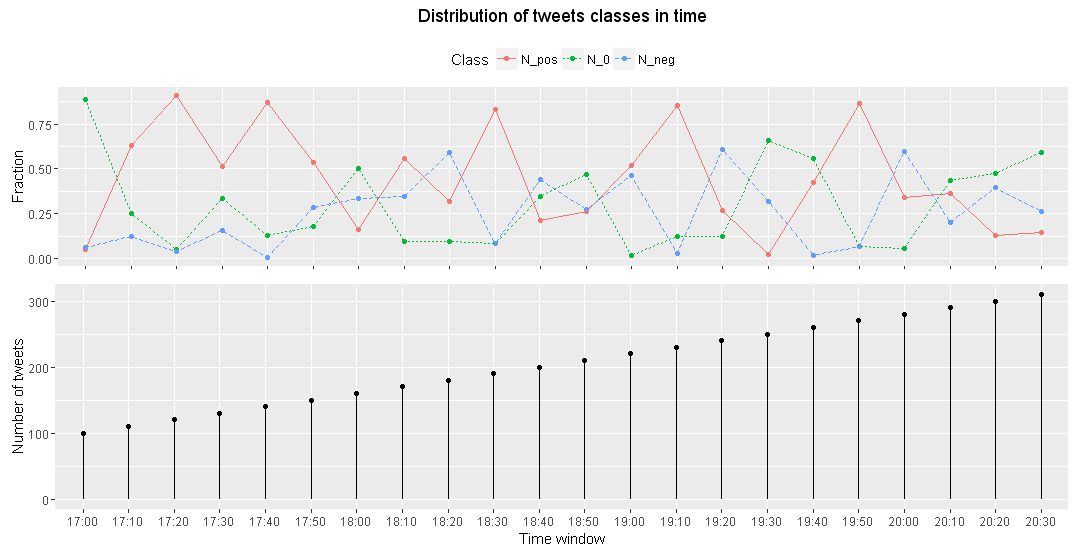
16:30 – 17:20 : 130 0.5/0.25/0.25

...

16:30 – 20:30: 1000 0.4/0.4/0.2

Каждая строка в файле hours.txt описывает распределение твитов в некотором часовом интервале.

В заключение, необходимо построить сдвоенный график (один под другим). На первом графике должна быть изображена динамика распределения для каждого класса (разбить по цветам и типам линий). На втором – рост количества твитов. По оси Х на каждом графике отложены концы временных окон. График должен выглядеть почти как на примере ниже (цветы и типы линий могут быть любыми, естественно).



**7\*. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ. Определение точности эмпирической оценки твитов.**

Для каждого твита, согласно Вашей классификации, возможно найти оценку (см. пункт 4). Для каждого слова можно найти твиты, в котором слово участвует и посчитать среднюю оценку / для этих твитов. Таким образом можно определить, участвует слово в положительных или отрицательных твитах, и насколько Ваша изначальная эмпирическая оценка была точна.  
Для каждого слова вычисляем среднюю оценку по тем твитам, в котором оно участвует, и сравниваем с изначальной эмпирической оценкой. Необходимо вывести в файл estimation\_check.txt 5 слов с наиболее сильным расхождением оценок и 5 слов с наиболее слабым расхождением. Также необходимо посчитать точность общей оценки - то есть сравнить среднюю оценку для каждого слова с изначальной и в тех случаях, где разница не превышает определенного порога (порог подберите сами), считаем что оценки совпали. Общая точность – отношение количества совпавших оценок к общему числу оценок/слов(\* 100%).

Пример:

Top-5 Closest:  
добрый 1 1.02  
злой -1 -1.1  
хороший 1 1.5  
плохой -1 -2  
президент 0 0.5

Top-5 Furthest:  
добрый 1 1.02  
злой -1 -1.1  
хороший 1 1.5  
плохой -1 -2  
президент 0 0.5

Estimation accuracy: 57%

Первая строка в файле estimation\_check.txt – заголовок “ Top-5 Closest:”. Далее в каждой строке описывается слово, затем через пробел его изначальная оценка, затем через пробел его средняя оценка по всем твитам в котором оно участвует. Далее в таком же формате описывается блок с самыми удаленными оценками “ Top-5 Furthest:”.

В обоих блоках слова должны быть упорядочены по разнице с изначальными оценками. То есть первое слово – наиболее близкое/далекое, далее второе близкое/далекое и т.д

Далее идёт строка “Estimation accuracy:” за которой через пробел нужно вывести общую точность изначальной классификации.

**8\*. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ. Определить слова с самой положительной и отрицательной окраской.**

В предыдущем пункте мы определили среднюю оценку для каждого слова исходя из твитов, в котором оно участвует.

Теперь найдем среди этих слов 5 с наиболее низкими значениями оценки и 5 с наиболее высокими значениями оценки.

Полученные результаты запишем в файл best\_worst.txt и отразим на гистограмме.

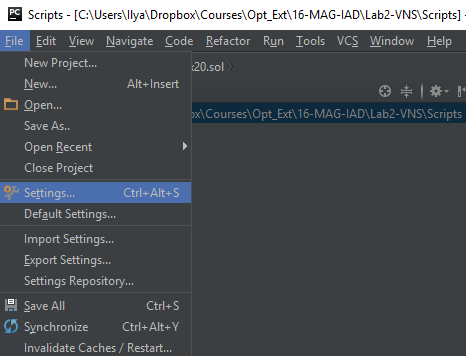
Top-5 Most Positive:  
стабильность 3.5  
будущее 3  
PutinTeam 2.5  
единство 2  
лучший 1.5

Top-5 Most Negative:  
бедность -4  
коррупция -3  
шурыгина -2  
военный -1.9  
оппозиция -1.5

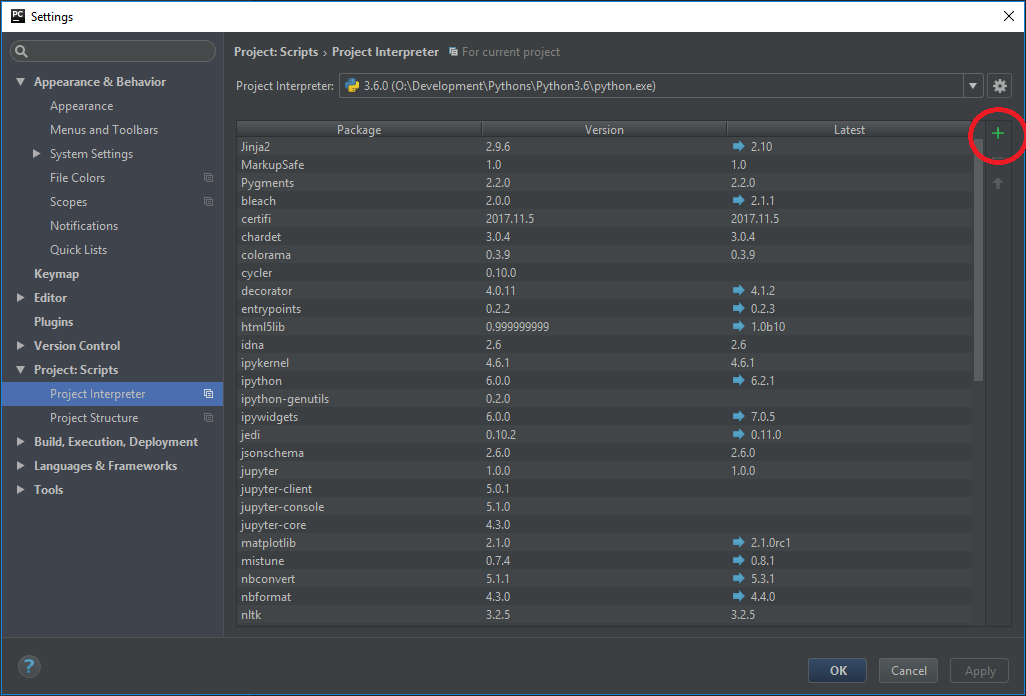
**Приложение 1. Установка модулей/библиотек в PyCharm.**

Для данной лабораторной будут полезны библиотеки NumPy, NLTK, MATPLOTLIB.

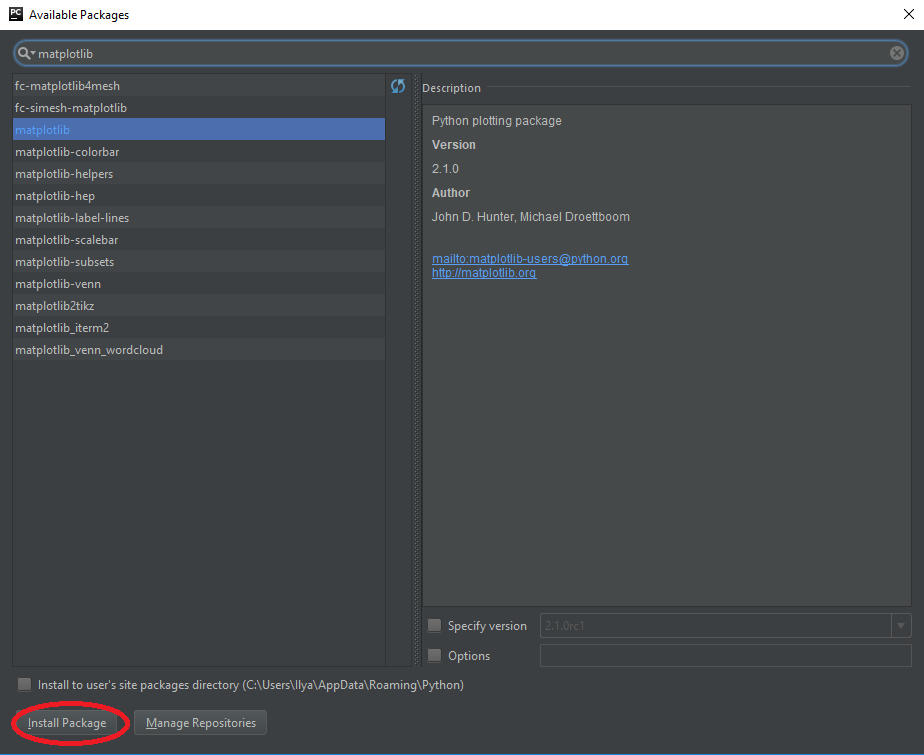
Шаг 1. Открываем меню File -> Settings



Шаг 2. Выбираем вкладку Project: \*\*\* (имя вашего проекта) -> Project Interpreter и нажимаем кнопу “+” справа



Шаг 3. В появившемся окне набираем имя нужного модуля, выбираем его в появившемся списке и нажимаем кнопку “Install Package”



Шаг 4. Далее ждём появления зелёной надписи о том, что модуль установлен, закрываем все открытые окна и можем использовать модуль в своем коде с помощью инструкции import.

